



TRABAJO DE GRADO
Opción Seminario-Diplomado.

Proyectos Crehana Seminario De Machine Learning

Corporación Universitaria Remington

Facultad de Ingenierías

Ingeniería de Sistemas

Estudiante: Juan Pablo Serna Quintero

Tutor: Juan Pablo Vélez Uribe

Opción de Trabajo de grado Seminario

2024



TRABAJO DE GRADO
Opción Seminario-Diplomado.

TRABAJO DE GRADO
Opción Seminario-Diplomado.

Tabla de contenido

Proyecto 1: Análisis contractual.....	4
Proyecto 2: Evaluación de modelos de machine learning	6
Proyecto 3: Fundamentos aplicados de machine learning.....	8
Ejercicio 1:.....	8
Ejercicio 2:.....	10
Ejercicio 3:.....	12
Proyecto 4: Innovación tecnológica con inteligencia artificial	13
Base de datos elegida	13
Preguntas de investigación.....	13
Pregunta seleccionada y columnas parametrizadas.....	14
Pasos a seguir y elección de IA.....	14
Representación gráfica de los resultados	15
Proyecto 5: Introducción a la ética en la inteligencia artificial.....	15
Proyecto 6: Introducción a la inteligencia artificial	17
Ejercicio 1:.....	18
Ejercicio 2:.....	19
Ejercicio 3:.....	26
Proyecto 6: Introducción a machine learning.....	30
Avance 1: Contexto y Problema a Resolver	30
Avance 2: Tipo de Predicción y Datos	30
Avance 3: Roles y Áreas Involucradas.....	30
Retos:	30
Proyecto 7: Machine learning aprendizaje supervisado.....	31

TRABAJO DE GRADO
Opción Seminario-Diplomado.

```
47
48  datos_modelo <- clientes_promocion[, variables]
49
50  # Dividir los datos en conjunto de entrenamiento y prueba
51  set.seed(123) # Fijar semilla para reproducibilidad
52  indices_entrenamiento <- sample(1:nrow(datos_modelo), 0.8 * nrow(datos_modelo))
53  datos_entrenamiento <- datos_modelo[indices_entrenamiento, ]
54  datos_prueba <- datos_modelo[-indices_entrenamiento, ]
55
56  # Instalar y cargar el paquete 'grf' para Causal Forest
57  install.packages("grf")
58  library(grf)
59
60  # Entrenar el modelo causal
61  modelo_causal <- causal_forest(grupo_promocion ~ ., data = datos_entrenamiento)
62
63  # Predecir el efecto causal sobre los datos de prueba
64  efecto_causal <- predict(modelo_causal, X = datos_prueba)
65
66  # Calcular el efecto medio del tratamiento
67  efecto_medio_tratamiento <- mean(efecto_causal$predictions)
68
69  print(paste("El efecto medio del tratamiento es:", efecto_medio_tratamiento))
70
```

TRABAJO DE GRADO Opción Seminario-Diplomado.

Proyecto 2: Evaluación de modelos de machine learning

```
Proyecto final - Evaluación de modelos de machine learning.ipynb X
C:\Users\juamp\Desktop>U>proyectos finales seminario>Proyecto final - evaluación de modelos de machine learning>Proyecto final - Evaluación de modelos de machine learning.ipynb
+ Code + Markdown | ▶ Run All | Clear All Outputs | Outline ...

import pandas as pd
from sklearn.model_selection import train_test_split,cross_val_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier

# AVANCE 1

# Cargar los datos desde el archivo CSV
data = pd.read_csv("data.csv")

# Revisar los primeros registros del DataFrame
print("Primeros registros del DataFrame:")
print(data.head())

# Información general sobre el DataFrame
print("\nInformación general sobre el DataFrame:")
print(data.info())

# Resumen estadístico de las variables numéricas
print("\nResumen estadístico de las variables numéricas:")
print(data.describe())

# Revisar valores nulos
print("\nValores nulos en el DataFrame:")
print(data.isnull().sum())

# Revisar valores perdidos (si es que hay algún tipo de marcador de valores perdidos)
# No hay marcadores explícitos de valores perdidos, pero podríamos buscar valores que podrían indicar datos faltantes (por ejemplo, 0 en variables que no deberían ser 0)
# En este caso, no parece haber valores que indiquen datos faltantes

# Revisar datos atípicos
print("\nDatos atípicos (valores extremadamente altos o bajos):")
# Podemos visualizar datos atípicos mediante gráficos de caja o histogramas para cada variable numérica
# Por simplicidad, solo mostraremos los histogramas para las variables numéricas
data.hist(figsize=(15, 10))
plt.show()

# Partición del Dataset en conjunto de entrenamiento y conjunto de prueba
X = data.drop("diagnosis", axis=1) # Características
y = data["diagnosis"] # Variable objetivo
```

TRABAJO DE GRADO Opción Seminario-Diplomado.

```
Proyecto final - Evaluación de modelos de machine learning.ipynb X
C:\Users\juand\Desktop>U>proyectos finales seminario>Proyecto final - evaluación de modelos de machine learning> Proyecto final - Evaluación de modelos de machine learning.ipynb
+ Code + Markdown | ▶ Run All | Clear All Outputs | Outline ...
▶ x
# Definimos el conjunto de entrenamiento y conjunto de prueba
X, y = data[:"diagnosis", axis=1] # Características
y = data["diagnosis"] # Variable objetivo

# Utilizamos train_test_split para dividir los datos en 80% para entrenamiento y 20% para prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Mostrar las dimensiones de los conjuntos de entrenamiento y prueba
print("\nDimensiones de los conjuntos de entrenamiento y prueba:")
print("Conjunto de entrenamiento - Características:", X_train.shape)
print("Conjunto de entrenamiento - Variable objetivo:", y_train.shape)
print("Conjunto de prueba - Características:", X_test.shape)
print("Conjunto de prueba - Variable objetivo:", y_test.shape)

# AVANCE 2

# Definimos una función para evaluar el modelo y mostrar las métricas
def evaluate_model(y_true, y_pred):
    # Calcular y mostrar la precisión del modelo
    accuracy = accuracy_score(y_true, y_pred)
    print("Precisión del modelo:", accuracy)

    # Calcular y mostrar la precisión, recall y F1-score del modelo
    precision = precision_score(y_true, y_pred)
    recall = recall_score(y_true, y_pred)
    f1 = f1_score(y_true, y_pred)
    print("Precisión:", precision)
    print("Recall:", recall)
    print("F1-score:", f1)

    # Matriz de confusión para evaluar el rendimiento del modelo en detalle
    confusion_mat = confusion_matrix(y_true, y_pred)
    print("Matriz de confusión:")
    print(confusion_mat)

# Supongamos que ya hemos entrenado un modelo y tenemos predicciones
# En este caso, asumiremos que 'y_pred' contiene las predicciones del modelo y 'y_test' contiene las etiquetas verdaderas del conjunto de prueba

# Evaluamos el modelo utilizando las métricas definidas
print("\nEvaluación del modelo en el conjunto de prueba:")
evaluate_model(y_test, y_pred)

# AVANCE 3

# Definimos los modelos
svm_model = SVC()
rf_model = RandomForestClassifier()
```

```
# Validación cruzada para evaluar el rendimiento de SVM
svm_scores = cross_val_score(svm_model, X_train, y_train, cv=5, scoring='accuracy')
print("Puntuaciones de validación cruzada para SVM:", svm_scores)
print("Precisión media de SVM:", svm_scores.mean())

# Validación cruzada para evaluar el rendimiento de Random Forest
rf_scores = cross_val_score(rf_model, X_train, y_train, cv=5, scoring='accuracy')
print("\nPuntuaciones de validación cruzada para Random Forest:", rf_scores)
print("Precisión media de Random Forest:", rf_scores.mean())

# Comparación de modelos y selección del mejor
if svm_scores.mean() > rf_scores.mean():
    print("\nSVM tiene mejor precisión media. Seleccionando SVM como el modelo final.")
    final_model = svm_model
else:
    print("\nRandom Forest tiene mejor precisión media. Seleccionando Random Forest como el modelo final.")
    final_model = rf_model
```

TRABAJO DE GRADO Opción Seminario-Diplomado.

Proyecto 3: Fundamentos aplicados de machine learning

Ejercicio 1:

```
Proyecto 1 - Ejercicios en el Notebook.ipynb X
C:\Users\juamp\Desktop>U>proyectos finales seminario>Proyecto final - fundamentos aplicados de Machine Learning>Proyecto 1 - Ejercicios en el Notebook.ipynb # Define una función max()
+ Code + Markdown | ▶ Run All | Clear All Outputs | Outline ...
Define una función generar_n_caracteres() que tome un entero n y devuelva el caracter multiplicado por n. Por ejemplo: generar_n_caracteres(5, 'x') debería dev

# Define una función max() que tome como argumento dos números y devuelva el mayor de ellos.
def maximo(a, b):
    if a > b:
        return a
    else:
        return b

# Define una función max_de_tres(), que tome tres números como argumentos y devuelva el mayor de ellos.
def max_de_tres(a, b, c):
    return maximo(maximo(a, b), c)

# Define una función que calcule la longitud de una lista o una cadena dada.
def longitud(lista_o_cadena):
    count = 0
    for _ in lista_o_cadena:
        count += 1
    return count

# Escribe una función que tome un carácter y devuelva True si es una vocal, de lo contrario devuelve False.
def es_vocal(caracter):
    vocales = 'aeiouAEIOU'
    return caracter in vocales

# Escribe una función sum() y una función multiplic() que sumen y multipliquen respectivamente todos los números de una lista.
def suma(lista):
    total = 0
    for num in lista:
        total += num
    return total

def multiplicacion(lista):
    total = 1
    for num in lista:
        total *= num
    return total

# Define una función inversa() que calcule la inversión de una cadena.
def inversa(cadena):
    return cadena[::-1]

# Define una función superposicion() que tome dos listas y devuelva True si tienen al menos 1 miembro en común o devuelva False de lo contrario.
def superposicion(lista1, lista2):
    for elem1 in lista1:
        for elem2 in lista2:
```



TRABAJO DE GRADO Opción Seminario-Diplomado.

```
Proyecto 1 - Ejercicios en el Notebook.ipynb X
C:\Users\juamp\Desktop>U>proyectos finales seminario>Proyecto final - fundamentos aplicados de Machine Learning>Proyecto 1 - Ejercicios en el Notebook.ipynb # Define una función max() que tome como argu
+ Code + Markdown Run All Clear All Outputs Outline ...
# Define una función inversa() que calcule la inversión de una cadena.
return cadena[::-1]

# Define una función superposición() que tome dos listas y devuelva True si tienen al menos 1 miembro en común o devuelva False de lo contrario.
def superposicion(lista1, lista2):
    for elem1 in lista1:
        for elem2 in lista2:
            if elem1 == elem2:
                return True
    return False

# Define una función generar_n_caracteres() que tome un entero n y devuelva el caracter multiplicado por n.
def generar_n_caracteres(n, caracter):
    return caracter * n

# Uso de la función max()
print(maximo(10, 5)) # Debería imprimir 10

# Uso de la función max de tres()
print(max_de_tres(10, 5, 20)) # Debería imprimir 20

# Uso de la función longitud()
print(longitud("Hola, mundo")) # Debería imprimir 11

# Uso de la función es_vocal()
print(es_vocal('a')) # Debería imprimir True
print(es_vocal('z')) # Debería imprimir False

# Uso de la función sum() y multip()
print(suma([1, 2, 3, 4])) # Debería imprimir 10
print(multiplicacion([1, 2, 3, 4])) # Debería imprimir 24

# Uso de la función inversa()
print(inversa("estoy probando")) # Debería imprimir "odnaborp yotse"

# Uso de la función superposicion()
print(superposicion([1, 2, 3], [3, 4, 5])) # Debería imprimir True
print(superposicion([1, 2, 3], [4, 5, 6])) # Debería imprimir False

# Uso de la función generar_n_caracteres()
print(generar_n_caracteres(5, "x")) # Debería imprimir "xxxxx"
```

TRABAJO DE GRADO Opción Seminario-Diplomado.

Ejercicio 2:

```
# define "X" como todo el conjunto de variables explicativas
# define "Y" como la variable a predecir (precios)

X = df.drop('variable_objetivo', axis=1)
y = df['variable_objetivo']

# usa "train_test_split" adicionando los parámetros test_size = 0.2 y random_state = 42

X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.2, random_state=42 ) # 4 argumentos

# crea una instancia para la regresión lineal
reg_all = LinearRegression()

# usa el método "fit" de la regresión con los datos de entrenamiento (tanto para X como para y)
reg_all.fit( X_train, y_train )

# usa el método "predict" de la regresión para hacer las predicciones sobre el conjunto X de entrenamiento
y_train_predict = reg_all.predict( X_train )

# calcula el rmse
# pasa los parámetros adecuados a "mean_squared_error". Recuerda que esta métrica se calcula
# usando la diferencia entre
rmse = (np.sqrt(mean_squared_error(y_train, y_train_predict)))

# calcula el R cuadrado
r2 = round(reg_all.score(X_train, y_train),2)

print("Desempeño del modelo para los datos de entrenamiento")
print("-----")
print('RMSE: {}'.format(rmse))
print('R2: {}'.format(r2))
print("\n")
```

TRABAJO DE GRADO Opción Seminario-Diplomado.

```
# repite el proceso para los datos de test

y_pred = reg_all.predict(X_test)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
r2 = round(reg_all.score(X_test, y_test), 2)

print("Desempeño del modelo para los datos de test")
print("-----")
print("RMSE: {}".format(rmse))
print("R2: {}".format(r2))
print("\n")
```

- Compara frente al modelo guiado durante el módulo que construiste usando una única variable. ¿Cómo cambian los pronósticos en test y entrenamiento?

Esperamos ver predicciones mejoradas en los datos de prueba al pasar de un modelo de regresión lineal simple a uno más complejo que utiliza todas las variables disponibles.

Visualiza tus predicciones:

```
# pasa los precios reales de test y los pronósticos para graficarlos
plt.scatter(y_test, y_pred) # dos argumentos

# estas 5 líneas de código no necesitas modificarlas
plt.xlabel("Precio Real ($1000)")
plt.ylabel("Precio Pronosticado ($1000)")
plt.xticks(range(0, int(max(y_test)),2))
plt.yticks(range(0, int(max(y_test)),2))
plt.title("Precios Reales vs. Precios Pronosticados de Casas")
```

Otras preguntas:

- Para qué predecir? Para Tomar decisiones informadas. Los pronósticos pueden proporcionar información valiosa para la toma de decisiones en una variedad de contextos, desde estrategias comerciales hasta políticas públicas.
- Investiga qué empresas utilizan pronóstico de precios Empresas minoristas: utilice la previsión de precios para optimizar las estrategias de precios, gestionar el inventario y planificar promociones.

Empresas de servicios financieros: Utilizan previsiones de precios para predecir movimientos en los mercados financieros, evaluar riesgos y tomar decisiones de inversión.

Empresas inmobiliarias: utilice pronósticos de precios para predecir las tendencias del mercado inmobiliario, valorar propiedades y tomar decisiones de inversión.

Empresas de transporte y logística: utilice previsiones de precios para planificar rutas, gestionar la demanda y optimizar la utilización de recursos.

Empresas de tecnología: utilice la previsión de precios para fijar los precios de productos y servicios y predecir la demanda de productos tecnológicos.

- Aplicabilidad de este modelo de Boston hoy? Si bien el modelo de Boston sigue siendo una herramienta valiosa para examinar las relaciones entre las características y los precios de la vivienda, su aplicabilidad directa a los mercados inmobiliarios modernos puede requerir consideración adicional y potencial.

TRABAJO DE GRADO Opción Seminario-Diplomado.

Ejercicio 3:

```
Proyecto 3 - Clasificación de tarjetas de credito.ipynb × Proyecto 2 - Prediciendo el Precio de una Casa.ipynb
C: > Users > juanp > Desktop > U > proyectos finales seminario > Proyecto final - fundamentos aplicados de Machine Learning > Proyecto 3 - Clasificación de tarjetas de credito.ipynb > Clase 11: In
+ Code + Markdown ▶ Run All ☰ Clear All Outputs | ☰ Outline ...
```

- Empieza por importar GridsearchCV

```
[53] # TODO: Importa el módulo de sklearn "GridSearchCV"
      from sklearn.model_selection import GridSearchCV
```

- Ahora define los parámetros a explorar
- Crea dos listas de python "tol" y "max_iter", con los valores 0.01, 0.001, 0.0001 para "tol", y con 150, 200, 300, 1000 para "max_iter"

```
[54] # TODO: Define las dos listas (dos líneas de código)
      tol = [0.01, 0.001, 0.0001]
      max_iter = [150, 200, 300, 1000]
```

```
[56] # Crea un diccionario con dos llaves donde tol y max_iter son las llaves y las listas de arriba sus valores correspondientes
      param_grid = dict(tol=tol, max_iter=max_iter)

      param_grid # imprime para ver el diccionario
```

```
... {'tol': [0.01, 0.001, 0.0001], 'max_iter': [150, 200, 300, 1000]}
```

- A continuación, crea una instancia de GridSearchCV con los siguientes parámetros:
 - estimator = logreg
 - cv = 5
 - Para el parámetro "param_grid" pasa el diccionario que creaste arriba

```
[57] # TODO: crea la instancia de GridSearchCV y guárdala como una variable
      grid_model = GridSearchCV(estimator=logreg, param_grid=param_grid, cv=5)
```

TRABAJO DE GRADO Opción Seminario-Diplomado.

```
Projecto 3 - Clasificación de tarjetas de credito.ipynb x Projecto 2 - Prediciendo el Precio de una Casa.ipynb
Users > juamp > Desktop > U > proyectos finales seminario > Proyecto final - fundamentos aplicados de Machine Learning > Proyecto 3 - Clasificación de tarjetas de credito.ipynb > Clase 11: Importa y entiende la base
+ Code + Markdown | ▶ Run All | Clear All Outputs | Outline ...
```

- Para este ejercicio, usarás una técnica conocida como "Validación Cruzada" (Cross Validation), por lo cuál no necesitarás de antemano separar los datos en entrenamiento y test (el algoritmo lo hace internamente)
- Es por esto que usarás todo el conjunto de datos X, en vez de X_train o X_test
- Empieza por reescalar X (creado durante las clases), usando "scaler.fit_transform"
- Luego usa X reescalado, al igual que la variable "y" (tu variable objetivo completa), para estimar todos los modelos y obtener los resultados. Guárdalos en la variable "grid_model_result"
 - Para esto, usa la función "fit" de tu instancia de GridSearchCV "grid_model"

```
# Usa "scaler" para reescalar X y guárdalo en una variable
rescaledX = scaler.fit_transform(X)

# Ajusta los datos a grid_model
grid_model_result = grid_model.fit(rescaledX, y)
```

```
[59]
```

- ¡Imprime los resultados! Este código ya está preparado para ti

```
# Summarize results
best_score, best_params = grid_model_result.best_score_, grid_model_result.best_params_

# Creating a dictionary to save the best results
best_models = {'Logistic': best_score}

print(f'Mejor resultado: {best_score}, usando {best_params}')
```

```
[59]
```

```
... Mejor resultado: 0.8535174018830001, usando {'max_iter': 150, 'tol': 0.01}
```

```
... Mejor resultado: 0.8535174018830001, usando {'max_iter': 150, 'tol': 0.01}
```

- Finalmente, responde en pocas líneas: ¿Es el modelo que acabas de estimar, mejor que el preparado en clase?
resultado modelo preparado en clase = 0.8584070796460177 resultado modelo nuevo = 0.8535174018830001 en este caso, el modelo preparado en clase con una precisión de aproximadamente 0.8584 es ligeramente mejor que el modelo nuevo con una precisión de aproximadamente 0.8535

Proyecto 4: Innovación tecnológica con inteligencia artificial

Base de datos elegida

Top 50 canciones del 2023

Preguntas de investigación.

- ¿Cuáles son las características musicales más comunes en las canciones exitosas del 2023?
- ¿Existe alguna correlación entre el género de la canción y su popularidad?
- ¿Qué características influyen más en la duración de una canción en el top 50?
- ¿Hay alguna tendencia temporal en las características musicales de las canciones exitosas?
- ¿Se puede predecir el éxito de una canción utilizando características como el tempo, la energía y la valencia?

TRABAJO DE GRADO Opción Seminario-Diplomado.

Pregunta seleccionada y columnas parametrizadas.

¿Se puede predecir el éxito de una canción utilizando características como el tempo, la energía y la valencia?

- tempo: El tempo de la canción en BPM.
- energy: La energía percibida de la canción.
- valence: La positividad de la canción.
- popularity: La popularidad de la canción en una escala de 0 a 100 (nuestro objetivo de predicción).

Pasos a seguir y elección de IA

Un modelo de regresión lineal o un modelo de bosque aleatorio pueden ser adecuados para este propósito, dada la naturaleza constante de la popularidad de las canciones.

```
import pandas as pd  Untitled-3  # Importar librerías necesarias  Untitled-1
1  # Importar librerías necesarias
2  import pandas as pd
3  from sklearn.model_selection import train_test_split
4  from sklearn.ensemble import RandomForestRegressor
5  from sklearn.metrics import mean_squared_error, r2_score
6
7  # Cargar la base de datos
8  data = pd.read_csv("top_50_2023.csv")
9
10 # Seleccionar características y objetivo
11 X = data[['tempo', 'energy', 'valence']]
12 y = data['popularity']
13
14 # Dividir los datos en conjuntos de entrenamiento y prueba
15 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
16
17 # Inicializar y entrenar el modelo de bosque aleatorio
18 model = RandomForestRegressor(n_estimators=100, random_state=42)
19 model.fit(X_train, y_train)
20
21 # Predecir la popularidad en el conjunto de prueba
22 y_pred = model.predict(X_test)
23
24 # Evaluar el rendimiento del modelo
25 mse = mean_squared_error(y_test, y_pred)
26 r2 = r2_score(y_test, y_pred)
27
28 print("Mean Squared Error:", mse)
29 print("R^2 Score:", r2)
```

TRABAJO DE GRADO
Opción Seminario-Diplomado.

Representación gráfica de los resultados

```
# Predecir la popularidad en el conjunto de prueba
y_pred = model.predict(X_test)

# Evaluar el rendimiento del modelo
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("Mean Squared Error:", mse)
print("R^2 Score:", r2)

# Graficar predicciones vs valores reales
plt.figure(figsize=(8, 6))
plt.scatter(y_test, y_pred, color='blue', alpha=0.5)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--', lw=2)
plt.xlabel('Popularidad Real')
plt.ylabel('Popularidad Predicha')
plt.title('Predicciones vs Valores Reales')
plt.grid(True)
plt.show()
```

Proyecto 5: Introducción a la ética en la inteligencia artificial

- a. ¿Debería la compañía invertir en corregir el sesgo, incluso si eso significa un retraso en la implementación y posiblemente afectar su posición en el mercado?

R/ La corrección de la distorsión en el algoritmo es de importancia crucial desde una perspectiva ética y responsabilidad social. Aunque la corrección sería un retraso en la implementación y, a corto plazo, podría influir en su posición en el mercado, una corrección ética fortalecerá la reputación de la compañía a largo plazo y su compromiso con la equidad en la atención médica.

- b. Si la compañía decide corregir el sesgo, es posible que mucha gente no logre obtener un diagnóstico de manera veloz o precisa, ¿vale la pena corregir el sesgo incluso si afecta negativamente a las personas que no se veían afectadas anteriormente?

R/ Aunque la corrección puede conducir a una reducción temporal en la velocidad o precisión del diagnóstico para ciertos grupos, no para perpetuar la discriminación y el riesgo de diagnósticos incorrectos para los grupos afectados. Es muy importante determinar la prioridad en la justicia y la precisión en el diagnóstico de efectividad comercial a corto plazo.

TRABAJO DE GRADO
Opción Seminario-Diplomado.

- c. ¿Cómo debería la compañía abordar las acusaciones de infracción de violación de la privacidad? ¿Deberían compensar a los pacientes cuyos datos se utilizaron sin permiso, y cómo podría hacerse esto sin crear un precedente peligroso?

R/ La Compañía debe tratar las acusaciones de violación de la protección de datos con transparencia y responsabilidad. Con respecto a la compensación, la compañía podría considerar ofrecer a los pacientes afectados servicios médicos gratuitos o descuentos en tratamientos futuros para remediar el daño causado.

- d. ¿Hasta qué punto debería la compañía revelar cómo funciona su algoritmo?

R/ La transparencia en el funcionamiento del algoritmo es necesaria para aumentar la confianza entre los trabajadores médicos y entre los pacientes. Aunque la compañía puede proteger ciertos aspectos de su propiedad intelectual, debe dar una descripción general de cómo se capacita el algoritmo, qué datos se usan y cómo se lleva a cabo la precisión.

- e. ¿Cómo pueden los líderes de la compañía equilibrar las demandas de todas estas áreas y asegurar que su tecnología no solo sea comercialmente exitosa sino también éticamente responsable?

R/ Los líderes de la compañía deben adoptar un enfoque proactivo para superar los problemas éticos y de privacidad relacionados con su tecnología. Al hacerlo, la compañía puede mantener su competitividad en el mercado mientras cumple con sus obligaciones éticas y sociales.



TRABAJO DE GRADO
Opción Seminario-Diplomado.

Proyecto 6: Introducción a la inteligencia artificial

TRABAJO DE GRADO
Opción Seminario-Diplomado.

Ejercicio 1:

```
from keras.models import load_model # TensorFlow is required for Keras to work
from PIL import Image, ImageOps # Install pillow instead of PIL
import numpy as np

# Disable scientific notation for clarity
np.set_printoptions(suppress=True)

# Load the model
model = load_model("keras_Model.h5", compile=False)

# Load the labels
class_names = open("labels.txt", "r").readlines()

# Create the array of the right shape to feed into the keras model
# The 'length' or number of images you can put into the array is
# determined by the first position in the shape tuple, in this case 1
data = np.ndarray(shape=(1, 224, 224, 3), dtype=np.float32)

# Replace this with the path to your image
image = Image.open("<IMAGE_PATH>").convert("RGB")

# resizing the image to be at least 224x224 and then cropping from the center
size = (224, 224)
image = ImageOps.fit(image, size, Image.Resampling.LANCZOS)

# turn the image into a numpy array
image_array = np.asarray(image)

# Normalize the image
normalized_image_array = (image_array.astype(np.float32) / 127.5) - 1

# Load the image into the array
data[0] = normalized_image_array

# Predicts the model
prediction = model.predict(data)
index = np.argmax(prediction)
class_name = class_names[index]
confidence_score = prediction[0][index]

# Print prediction and confidence score
print("Class:", class_name[2:], end="")
print("Confidence Score:", confidence_score)
```

TRABAJO DE GRADO Opción Seminario-Diplomado.

Ejercicio 2:

```
Plantilla_NN_CIFAR100.ipynb X
C: > Users > juanp > Desktop > U > proyectos finales seminario > Proyecto final - introduccion a la inteligencia artificial > Plantilla_NN_CIFAR100.ipynb > empty cell
+ Code + Markdown | ▶ Run All ☰ Clear All Outputs | ☰ Outline ...
```

Importando librerías

```
import tensorflow as tf
from tensorflow import keras
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Flatten, Dense
```

[18]

Carga de Datos

Importamos las librerías que utilizaremos para cargar los datos para la red neuronal:

```
from tensorflow.keras.datasets import cifar100
```

[10]

Corre la siguiente línea para cargar tu dataset

```
(train_images, train_labels), (test_images, test_labels) = cifar100.load_data()
```

[14]

Revisa las dimensiones de tus datos

```
print("Dimensiones de los datos de entrenamiento:")
print("Imágenes de entrenamiento:", train_images.shape)
print("Etiquetas de entrenamiento:", train_labels.shape)

print("\nDimensiones de los datos de prueba:")
print("Imágenes de prueba:", test_images.shape)
print("Etiquetas de prueba:", test_labels.shape)
```

[15]

TRABAJO DE GRADO Opción Seminario-Diplomado.

Plantilla_NN_CIFAR100.ipynb X

C: > Users > juanp > Desktop > U > proyectos finales seminario > Proyecto final - introduccion a la inteligencia artificial > Plantilla_NN_CIFAR100.ipynb > empty cell

+ Code + Markdown | Run All Clear All Outputs Outline ...

```
... Dimensiones de los datos de entrenamiento:  
Imágenes de entrenamiento: (50000, 32, 32, 3)  
Etiquetas de entrenamiento: (50000, 1)  
  
Dimensiones de los datos de prueba:  
Imágenes de prueba: (10000, 32, 32, 3)  
Etiquetas de prueba: (10000, 1)
```

Abre el primer registro de train

```
print("Primer registro del conjunto de datos de entrenamiento:")  
print("Etiqueta del primer registro:", train_labels[0])
```

[16]


```
... Primer registro del conjunto de datos de entrenamiento:  
Etiqueta del primer registro: [19]
```

Visualiza la primer imagen del dataset

```
plt.imshow(train_images[0])  
plt.axis('off') # Para eliminar los ejes  
plt.show()
```

[17]

...



TRABAJO DE GRADO
Opción Seminario-Diplomado.

Modelo

Crea tu red neuronal con una forma de 32*32 px

```
model = Sequential([
    Flatten(input_shape=(32, 32, 3)),
    Dense(128, activation='relu'),
    Dense(100, activation='softmax')
])
```

```
model.summary()
```

[19]

... Model: "sequential"

Layer (type)	Output Shape	Param #
flatten (Flatten)	(None, 3072)	0
dense (Dense)	(None, 128)	393344
dense_1 (Dense)	(None, 100)	12900

=====
Total params: 406244 (1.55 MB)

Trainable params: 406244 (1.55 MB)

Non-trainable params: 0 (0.00 Byte)

Transformación de datos

Ejecuta la carga de datos para el train y test

```
(x_train, y_train), (x_test, y_test) = cifar100.load_data()
```

[20]

TRABAJO DE GRADO Opción Seminario-Diplomado.

```
Plantilla_NN_CIFAR100.ipynb X
C: > Users > juanp > Desktop > U > proyectos finales seminario > Proyecto final - introduccion a la inteligencia artificial > Plantilla_NN_CIFAR100.ipynb > empty cell
+ Code + Markdown | ▶ Run All | Clear All Outputs | Outline ...
```

Transformación de datos

Ejecuta la carga de datos para el train y test

```
(x_train, y_train), (x_test, y_test) = cifar100.load_data()
```

[20]

Revisa las dimensiones de los datos

```
print("Dimensiones de los datos de entrenamiento:")
print("Imágenes de entrenamiento:", train_images.shape)
print("Etiquetas de entrenamiento:", train_labels.shape)

print("\nDimensiones de los datos de prueba:")
print("Imágenes de prueba:", test_images.shape)
print("Etiquetas de prueba:", test_labels.shape)
```

[21]

```
... Dimensiones de los datos de entrenamiento:
Imágenes de entrenamiento: (50000, 32, 32, 3)
Etiquetas de entrenamiento: (50000, 1)

Dimensiones de los datos de prueba:
Imágenes de prueba: (10000, 32, 32, 3)
Etiquetas de prueba: (10000, 1)
```

Haz un reshape para x_test y x_train

```
x_train = train_images.reshape((train_images.shape[0], 32, 32, 3))
x_test = test_images.reshape((test_images.shape[0], 32, 32, 3))
```

[22]

Importa la librería "to_categorical" de tensorflow

```
from tensorflow.keras.utils import to_categorical
```


TRABAJO DE GRADO Opción Seminario-Diplomado.

```
Plantilla_NN_CIFAR100.ipynb X
C: > Users > juanp > Desktop > U > proyectos finales seminario > Proyecto final - introduccion a la inteligencia artificial > Plantilla_NN_CIFAR100.ipynb
+ Code + Markdown | ▶ Run All | ☰ Clear All Outputs | ☰ Outline ...
Iteraciones de las épocas

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model.fit(x_train, y_train_categorical, epochs=10)

[28]
...
Epoch 1/10
1563/1563 [=====] - 7s 3ms/step - loss: 8.8925 - accuracy: 0.0088
Epoch 2/10
1563/1563 [=====] - 4s 3ms/step - loss: 4.6059 - accuracy: 0.0085
Epoch 3/10
1563/1563 [=====] - 5s 3ms/step - loss: 4.6058 - accuracy: 0.0096
Epoch 4/10
1563/1563 [=====] - 5s 3ms/step - loss: 4.6058 - accuracy: 0.0088
Epoch 5/10
1563/1563 [=====] - 5s 3ms/step - loss: 4.6058 - accuracy: 0.0090
Epoch 6/10
1563/1563 [=====] - 5s 3ms/step - loss: 4.6059 - accuracy: 0.0084
Epoch 7/10
1563/1563 [=====] - 5s 3ms/step - loss: 4.6059 - accuracy: 0.0086
Epoch 8/10
1563/1563 [=====] - 6s 4ms/step - loss: 4.6058 - accuracy: 0.0081
Epoch 9/10
1563/1563 [=====] - 9s 6ms/step - loss: 4.6058 - accuracy: 0.0082
Epoch 10/10
1563/1563 [=====] - 6s 4ms/step - loss: 4.6059 - accuracy: 0.0084
...
<keras.src.callbacks.History at 0x7c45f8917d00>

Evaluación

Evalua el set de train y el set de test

▶
train_loss, train_accuracy = model.evaluate(x_train, y_train_categorical, verbose=0)
print("Pérdida en el conjunto de entrenamiento:", train_loss)
print("Precisión en el conjunto de entrenamiento:", train_accuracy)

test_loss, test_accuracy = model.evaluate(x_test, y_test_categorical, verbose=0)
print("\nPérdida en el conjunto de prueba:", test_loss)
print("Precisión en el conjunto de prueba:", test_accuracy)

r 201
```

TRABAJO DE GRADO
Opción Seminario-Diplomado.

Evaluación

Evalua el set de train y el set de test

```
train_loss, train_accuracy = model.evaluate(x_train, y_train_categorical, verbose=0)
print("Pérdida en el conjunto de entrenamiento:", train_loss)
print("Precisión en el conjunto de entrenamiento:", train_accuracy)

test_loss, test_accuracy = model.evaluate(x_test, y_test_categorical, verbose=0)
print("\nPérdida en el conjunto de prueba:", test_loss)
print("Precisión en el conjunto de prueba:", test_accuracy)
```

[29]

```
... Pérdida en el conjunto de entrenamiento: 4.605103969573975
Pérdida en el conjunto de entrenamiento: 0.01001999992877245

Pérdida en el conjunto de prueba: 4.605213165283203
Pérdida en el conjunto de prueba: 0.009999999776482582
```

TRABAJO DE GRADO Opción Seminario-Diplomado.

Ejercicio 3:

```
Plantilla_PLN_Spotify.ipynb
C: > Users > juanp > Desktop > U > proyectos finales seminario > Proyecto final - introduccion a la inteligencia artificial > Plantilla_PLN_Spotify.ipynb > M+empty cell
+ Code + Markdown | ▶ Run All | Clear All Outputs | Outline ...
```

Importa el archivo de datos de Spotify

```
consumer_key = ''
consumer_secret = ''
access_token = ''
access_token_secret = ''

import tweepy
import pandas as pd

auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)
api = tweepy.API(auth)

twitter_users = []
tweet_time = []
tweet_string = []
```

[14]

Visualiza los primeros 5 registros

```
for tweet in tweepy.Cursor(api.search_tweets, q='spotify', count=1000).items(1000):
    if (not tweet.retweeted) and ('RT @' not in tweet.text):
        if tweet.lang == 'es':
            twitter_users.append(tweet.user.name)
            tweet_time.append(tweet.created_at)
            tweet_string.append(tweet.text)
```

[]

Convierte los tweets a listas

```
df = pd.DataFrame({'name':twitter_users, 'time':tweet_time, 'tweet':tweet_string })
df.shape
df.to_csv('tweets_spotify.csv')
data = df['tweet'].to_list()
```

[]

TRABAJO DE GRADO Opción Seminario-Diplomado.

Plantilla_PLN_Spotify.ipynb

C: > Users > juanp > Desktop > U > proyectos finales seminario > Proyecto final - introduccion a la inteligencia artificial > Plantilla_PLN_Spotify.ipynb

+ Code + Markdown ▶ Run All ☰ Clear All Outputs ☰ Outline ...

Ejecuta el patrón

```
pattern = r'''(?x)
          (?:[A-Z]\.)+      # Flag para iniciar el modo verbose
          | \w+(?:-\w+)*    # Hace match con abreviaciones como U.S.A.
          | \d+(?:\.\d+)?%? # Hace match con palabras que pueden tener un guión interno
          | \.\.\.         # Hace match con dinero o porcentajes como $15.5 o 100%
          | [\.,;'"'():-_\] # Hace match con puntos suspensivos
          | [\.,;'"'():-_\] # Hace match con signos de puntuación
          ...
'''
```

[]

Importa nltk, y el tokenizador

```
import nltk
nltk.download('punkt')
from nltk import word_tokenize
```

[]

Genera un vector vacío y un ciclo for que:

- Convierta a minúsculas las palabras
- Tokenize los tweets y los pase por el patrón
- Pegue los tokens en el vector vacío

```
texto = []

for x in range(0, len(data)):
    token_1 = data[x].lower()
    token_2 = nltk.regexp_tokenize(token_1, pattern)
    texto.append(token_2)
```

[]

+ Code

"Aplana" la lista de listas para que se convierta a **una sola** lista

```
flatten = [w for l in texto for w in l]
```

[]

TRABAJO DE GRADO Opción Seminario-Diplomado.

```
Plantilla_PLN_Spotify.ipynb •
C: > Users > juanp > Desktop > U > proyectos finales seminario > Proyecto final - introducción a la inteligencia artificial > Plantilla_PLN_Spotify.ipynb > empty cell
+ Code + Markdown | Run All Clear All Outputs Outline ...
```

Importa la librería de string y convierte los signos de puntuación a una lista

```
import string
puntuacion = list(string.punctuation)
```

Descarga la lista de stopwords en español

```
nlk.download('stopwords')
stop_words_n = nltk.corpus.stopwords.words('spanish')
```

Genera una nueva variable donde pase por un ciclo for para eliminar las stop words

```
df_2 = [w for w in flatten if w not in stop_words_n]
```

Genera una nueva variable donde pase por un ciclo for para eliminar los signos de puntuación

```
df_3 = [w for w in df_2 if w not in puntuacion]
```

Calcula con la función FreqDist la frecuencia de las palabras y almacenalos en una variable

```
freq_words = nltk.FreqDist(df_3)
```

Imprime las 20 palabras más comunes

```
freq_words.most_common(20)
```

TRABAJO DE GRADO Opción Seminario-Diplomado.

```
Plantilla_PLN_Spotify.ipynb •
C: > Users > juanp > Desktop > U > proyectos finales seminario > Proyecto final - introduccion a la inteligencia artificial > Plantilla_PLN_Spotify.ipynb > M+empty cell
+ Code + Markdown | ▶ Run All ≡ Clear All Outputs | ≡ Outline ...
```

Si crees que es necesario genera una lista de palabras a omitir

```
omitir_palabras = ['spotify']
```

Corre un ciclo for para omitir las palabras de nuestra lista

```
df_4 = [w for w in df_3 if w not in omitir_palabras]
```

Calcula nuevamente las frecuencias de las palabras

```
freq_words = nltk.FreqDist(df_4)
```

Imprime las 20 palabras más comunes

```
freq_words.most_common(20)
```

Importa las librerías para generar la nube de palabras y poder visualizarlas

```
from wordcloud import WordCloud
import matplotlib.pyplot as plt
```

```
Genera la gráfica de la nube de palabras
```

```
wordcloud = WordCloud(background_color='white', collocations=False, max_words=30).fit_words(freq_words)
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```

+ Code + Markdown

TRABAJO DE GRADO
Opción Seminario-Diplomado.

Proyecto 6: Introducción a machine learning

Avance 1: Contexto y Problema a Resolver

En un mundo cada vez más digitalizado, las empresas buscan optimizar sus ventas a través de redes sociales. El problema radica en la dificultad de gestionar eficientemente las interacciones con los clientes potenciales en estas plataformas. La creación de un asistente virtual de ventas por redes tiene como objetivo mejorar la atención al cliente, la personalización de las interacciones y, en última instancia, aumentar las conversiones de ventas.

Avance 2: Tipo de Predicción y Datos

El tipo de predicción aquí podría centrarse en la probabilidad de conversión de un cliente potencial en función de su interacción con el asistente virtual. Los datos podrían incluir interacciones históricas en redes sociales, como mensajes, respuestas, tiempos de respuesta, preferencias del cliente, historial de compras si está disponible y comportamiento en línea relacionado con la marca.

Avance 3: Roles y Áreas Involucradas

1. **Desarrollador de Machine Learning:** Encargado de crear y entrenar el modelo de machine learning para analizar datos de interacciones y predecir la probabilidad de conversión.
2. **Expertos en Ventas y Marketing:** Proporcionan información sobre el flujo de ventas, las tácticas efectivas en redes sociales y las métricas relevantes para evaluar el rendimiento del asistente virtual.
3. **Ingeniero de Datos:** Responsable de recopilar, limpiar y preparar los datos provenientes de las redes sociales y otros sistemas de la empresa para su uso en el modelo de machine learning.
4. **Especialista en Experiencia del Usuario (UX):** Colabora en el diseño de la interfaz del asistente virtual para garantizar una experiencia fluida y satisfactoria para los usuarios.

Retos:

- **Calidad de los Datos:** Los datos de interacciones en redes sociales pueden ser ruidosos y variados. Limpiar y estructurar estos datos puede ser desafiante.
- **Desarrollo del Modelo:** Crear un modelo preciso que pueda predecir con confiabilidad la probabilidad de conversión basándose en datos de redes sociales.

TRABAJO DE GRADO Opción Seminario-Diplomado.

- **Interfaz y Experiencia del Usuario:** Diseñar una interfaz intuitiva y atractiva para el asistente virtual que garantice una interacción positiva y efectiva con los clientes potenciales.

Este proyecto podría agregar un valor significativo al mejorar la eficiencia de las ventas en redes sociales y aumentar la tasa de conversión al ofrecer una interacción más personalizada y efectiva con los clientes.

Proyecto 7: Machine learning aprendizaje supervisado

```
Proyecto final Machine Learning Aprendizaje supervisado.ipynb X
C:\Users\juangp\Desktop>U>proyectos finales seminario>Proyecto final - machine learning aprendizaje supervisado> Proyecto final Machine Learning Aprendizaje supervisado.jp
+ Code + Markdown | ▶ Run All | Clear All Outputs | Outline ...

## PARTE 1
# Importar librerías necesarias
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.tree import DecisionTreeRegressor, DecisionTreeClassifier
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.ensemble import RandomForestRegressor, RandomForestClassifier
import tensorflow as tf

# Cargar los datos
datos = pd.read_csv('bd_clientes_2020.csv')

# Mostrar las primeras filas del dataframe para entender la estructura de los datos
print(datos.head())

# Identificar variables continuas y categóricas
variables_continuas = datos.select_dtypes(include=['float64', 'int64']).columns
variables_categoricas = datos.select_dtypes(include=['object']).columns

# Calcular estadísticas descriptivas para variables continuas
estadisticas_continuas = datos.groupby(['segmento', 'mes'])[variables_continuas].agg(['mean', 'std', 'sum'])

# Calcular frecuencia para variables categóricas
frecuencia_categoricas = {}
for col in variables_categoricas:
    frecuencia_categoricas[col] = datos[col].value_counts()

# Mostrar resultados
print("Variables continuas:")
print(estadisticas_continuas)
print("\nVariables categóricas:")
for col, frecuencia in frecuencia_categoricas.items():
    print(f"\n{col}:")
    print(frecuencia)

## PARTE 2

# Dividir los datos en conjunto de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(datos[variables_continuas], datos['target'], test_size=0.2, random_state=42)

# Regresión lineal
regresion_lineal = LinearRegression()
regresion_lineal.fit(X_train, y_train)
```

TRABAJO DE GRADO Opción Seminario-Diplomado.

```
Proyecto final Machine Learning Aprendizaje supervisado.ipynb X
C: > Users > juanp > Desktop > U > proyectos finales seminario > Proyecto final - machine learning aprendizaje supervisado > Proyecto final Mac
+ Code + Markdown | ▶ Run All | Clear All Outputs | Outline ...
LinearRegression()
regresion_lineal.fit(X_train, y_train)
predicciones_rl = regresion_lineal.predict(X_test)
mae_rl = mean_absolute_error(y_test, predicciones_rl)
mse_rl = mean_squared_error(y_test, predicciones_rl)
rmse_rl = mse_rl ** 0.5

# Árbol de decisión de regresión
arbol_decision_regresion = DecisionTreeRegressor()
arbol_decision_regresion.fit(X_train, y_train)
predicciones_adr = arbol_decision_regresion.predict(X_test)
mae_adr = mean_absolute_error(y_test, predicciones_adr)
mse_adr = mean_squared_error(y_test, predicciones_adr)
rmse_adr = mse_adr ** 0.5

# Regresión logística
log_reg = LogisticRegression()
log_reg.fit(X_train, y_train)
predicciones_log_reg = log_reg.predict(X_test)

# Evaluación de modelos
print("Métricas de Regresión Lineal:")
print("MAE:", mae_rl)
print("MSE:", mse_rl)
print("RMSE:", rmse_rl)

print("\nMétricas de Árbol de Decisión de Regresión:")
print("MAE:", mae_adr)
print("MSE:", mse_adr)
print("RMSE:", rmse_adr)

print("\nAccuracy de Regresión Logística:", log_reg.score(X_test, y_test))

# Visualización de resultados
plt.scatter(y_test, predicciones_rl)
plt.xlabel('Valores reales')
plt.ylabel('Predicciones')
plt.title('Regresión Lineal: Valores reales vs Predicciones')
plt.show()

## PARTE 3

# Árbol de decisión para clasificación
arbol_decision_clasificacion = DecisionTreeClassifier()
arbol_decision_clasificacion.fit(X_train, y_train)
accuracy_adc = arbol_decision_clasificacion.score(X_test, y_test)
```

TRABAJO DE GRADO Opción Seminario-Diplomado.

```
Proyecto final Machine Learning Aprendizaje supervisado.ipynb X
C: > Users > juanp > Desktop > U > proyectos finales seminario > Proyecto final - machine learning aprendizaje supervisado > Proyecto f
+ Code + Markdown | ▶ Run All | Clear All Outputs | Outline ...
▶ para clasificación
arbol_decision_clasificacion = DecisionTreeClassifier()
arbol_decision_clasificacion.fit(X_train, y_train)
accuracy_adc = arbol_decision_clasificacion.score(X_test, y_test)

# Random Forest para datos continuos
param_grid = {'n_estimators': [50, 100, 200],
              'max_depth': [None, 10, 20]}
rf_reg = RandomForestRegressor()
rf_grid_reg = GridSearchCV(rf_reg, param_grid, cv=5)
rf_grid_reg.fit(X_train, y_train)
best_rf_reg = rf_grid_reg.best_estimator_
importancias_rf_reg = best_rf_reg.feature_importances_

# Random Forest para datos categóricos
rf_clas = RandomForestClassifier()
rf_grid_clas = GridSearchCV(rf_clas, param_grid, cv=5)
rf_grid_clas.fit(X_train, y_train)
best_rf_clas = rf_grid_clas.best_estimator_
importancias_rf_clas = best_rf_clas.feature_importances_

# Mostrar importancia de variables
print("\nImportancia de variables en Random Forest para datos continuos:")
for i, var in enumerate(variables_continuas):
    print(f"{var}: {importancias_rf_reg[i]}")

print("\nImportancia de variables en Random Forest para datos categóricos:")
for i, var in enumerate(variables_categoricas):
    print(f"{var}: {importancias_rf_clas[i]}")

## PARTE 4

# Construir el modelo de red neuronal para regresión
model = tf.keras.Sequential([
    tf.keras.layers.Dense(64, activation='relu', input_shape=(X_train.shape[1],)),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dense(1)
])

# Compilar el modelo
model.compile(optimizer='adam', loss='mse', metrics=['mae', 'mse'])

# Entrenar el modelo
model.fit(X_train, y_train, epochs=100, batch_size=32, validation_data=(X_test, y_test))

# Evaluación del modelo
loss, mae, mse = model.evaluate(X_test, y_test)
```

TRABAJO DE GRADO
Opción Seminario-Diplomado.

```
# Evaluación del modelo  
loss, mae, mse = model.evaluate(X_test, y_test)  
print("Loss:", loss)  
print("MAE:", mae)  
print("MSE:", mse)
```